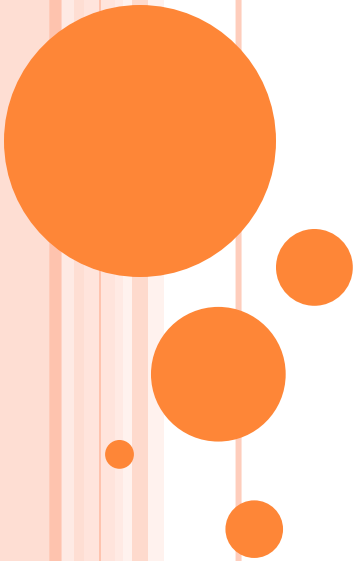
Primena metaheuristike mravljih kolonija za rešavanje problema pakovanja



Matematički fakultet, 2018.

Problem pakovanja (Bin Packing Problem - BPP)

- Problem pakovanja pripada klasi problema sečenja i pakovanja (*cutting and packing problems-CPP*). Opštu strukturu ove grupe problema opisao je Dyckhoff (1990) .

- Zadatak CPP: Raspolažemo kombinacijom grupe malih objekata (paketa) i grupe velikih objekata (skladišta). Cilj je optimalno dodeliti svaki od malih objekata nekom od velikih objekata (odnosno, spakovati pakete u skladišta), poštujući određena, unapred zadata pravila.

- Primeri CPP: problem ranca (*knapsack problem*), problem utovara prtljaga (*vehicle loading problem*), problem raspoređivanja zadataka na procesore (*multiprocessor sheduling problem*), problem raspoređivanja kapitala (*multi-period capital budgeting problem*), itd.

Problem pakovanja (Bin Packing Problem - BPP)

- BPP spada u jednostavnije probleme iz klase CPP
- Zadatak BPP: Dat je skup paketa sa pridruženim veličinama koje treba spakovati u skladišta unapred zadatog kapaciteta, tako da je broj iskorišćenih skladišta minimalan.
- BPP pripada klasi NP teških problema (Garey i Johnson, 1979).
- Egzaktne metode primenjene na BPP efikasno rešavaju instance problema manjih dimenzija, dok se za instance većih dimenzija koriste heuristike i metaheuristike.
- Najbolje rezultate daju hibridne metaheurističke metode

Definicija BPP

Ulazni podaci:

- konačan skup paketa: $L = \{1,\ldots,n\}$
- veličine paketa: l_j , $j= 1,\ldots,n$
- broj raspoloživih skladišta: N
- kapacitet svakog od skladišta: C

(pretpostavka je da su kapaciteti skladišta jednaki)

Cilj: spakovati pakete u što manji broj otvorenih skladišta, bez prekoračenja njihovih kapaciteta.

Matematička formulacija BPP

Matematička fomulacija problema koristi binarne promenljive:

$$x_{ij} = \begin{cases} 1, & ako\ je\ paket\ j\ u\ skladištu\ i \\ 0, & inače \end{cases} \qquad y_i = \begin{cases} 1, & ako\ je\ skladište\ i\ upotrebljeno \\ 0, & inače \end{cases}$$

Minimizujemo funkciju $z = \sum_{i=1}^{N} y_i$

Pri uslovima:

$$\sum_{j=1}^{n} l_j x_{ij} \leq C y_i , \qquad \forall\, i = 1,2, \dots, N$$

$$\sum_{i=1}^{N} x_{ij} = 1, \qquad \forall\ j = 1,2, \dots, n$$

$$x_{ij} \in \{0,1\}, \forall\ i,j$$

$$y_i \in \{0,1\}, \qquad \forall i$$

Primene problema pakovanja i njegovih varijanti

- ***Problem utovara prtljaga–*** minimizujemo broj vozila koja prevoze pakete određenih veličina do zadatih destinacija; ovde se dodatno mogu minimizovati i troškovi transporta.

- ***Problem pakovanja na traci*** - pakovanje 2D pravougaonika na traci fiksnih dimenzija u cilju minimizacije iskorišćene trake. Pravougaonici se ne smeju preklapati, a moguće je rotirati pravougaonike.

- ***Primene u papirnoj, čeličnoj, drvnoj industriji*** –cilj je iseći materijal na delove određene veličine, tako da se minimizuje materijal koji otpada (ovde je reč o 2D ili 3D problemu pakovanja).

- ***Neki problemi koji su su se svode na problem pakovanja,*** npr. VLSI dizajn (Very Large Scale Integration), tj. kreiranje integrisanih kola kombinovanjem velikog broja tranzistora u jedan čip (VLSI uređaj), planiranje procesorskog vremena, problemi iz ekonomije, npr. upravljanje kapitalom...

Dosadašnje metode rešavanje problema BPP

- **Greedy (pohlepne) heuristike – FFD, BFD, WFD, NFD**, (Dyckhoff , 1990)

Zajednička ideja: paketi seređaju u nerastućem poretku (po veličini), a koriste različite strategije

FFD (First Feasible Descreasing)– paket se smešta u prvo skladište u koje može stati.

BFD (Best Feasible Descreasing)– paket se smešta u prvo skladište u koje može stati, a koje ima najmanje preostalog slobodnog prostora.

WFD (Worst Feasible Descreasing) – paket se smešta u prvo skladište u koje može stati, a koje ima najviše preostalog slobodnog prostora.

NFD (Next Feasible Descreasing)– paketi se smeštaju u otvoreno skladište dok god mogu u njega stati.

Heuristike se zaustavljaju kada su svi paketi raspoređeni.

Heuristika FFD

(paket se smešta u prvo skladište u koje može stati)

Algoritam 2.1: Pseudo kod – FFD

```
Poređaj pakete nerastuće;
Ponavljaj:
        For svaki paket do
                For svako skladište do
                    If paket može stati u skladište
                       Spakuj paket;
                       Pređi na sledeći paket;
                    End if
                End for
            If paket ne može stati u skladište
               Otvori novo skladište;
               Spakuj paket u njega:
            End if
        End for
Sve dok: Nisu svi paketi spakovani;
Izlaz:   Paketi spakovani u skladišta.
```

Heuristika BFD

(paket se smešta u prvo skladište u koje može stati, a koje ima najmanje preostalog slobodnog prostora)

Algoritam 2.4: Pseudo kod – WFD

```
Poređaj pakete nerastuće;
Ponavljaj:
        For svaki paket do
                For svako skladište do
                Izračunaj preostali prostor u skladištu;
                   If skladište ima  min slobodnog prostora
                     i paket može stati u njega
                            Stavi paket;
                            Izračunaj preostali slobodan prostor u skladištu;
                   End if
                End for
           If paket ne može stati u skladište
              Otvori novo skladište;
              Stavi paket u njega:
           End if
        End for
Sve dok: Nisu svi paketi raspoređeni;
Izlaz:   Paketi raspoređeni u skladišta.
```

Heuristika WFD

(paket se smešta u prvo skladište u koje može stati, a koje ima najviše preostalog slobodnog prostora)

Algoritam 2.4: Pseudo kod – WFD

```
Poređaj pakete nerastuće;
Ponavljaj:
        For svaki paket do
                For svako skladište do
                Izračunaj preostali prostor u skladištu;
                   If skladište ima max slobodnog prostora
                     i paket može stati u njega
                            Stavi paket;
                            Izračunaj preostali slobodan prostor u skladištu;
                   End if
                End for
             If paket ne može stati u skladište
                Otvori novo skladište;
                Stavi paket u njega:
             End if
        End for
Sve dok: Nisu svi paketi raspoređeni;
Izlaz:   Paketi raspoređeni u skladišta.
```

Heuristika NFD

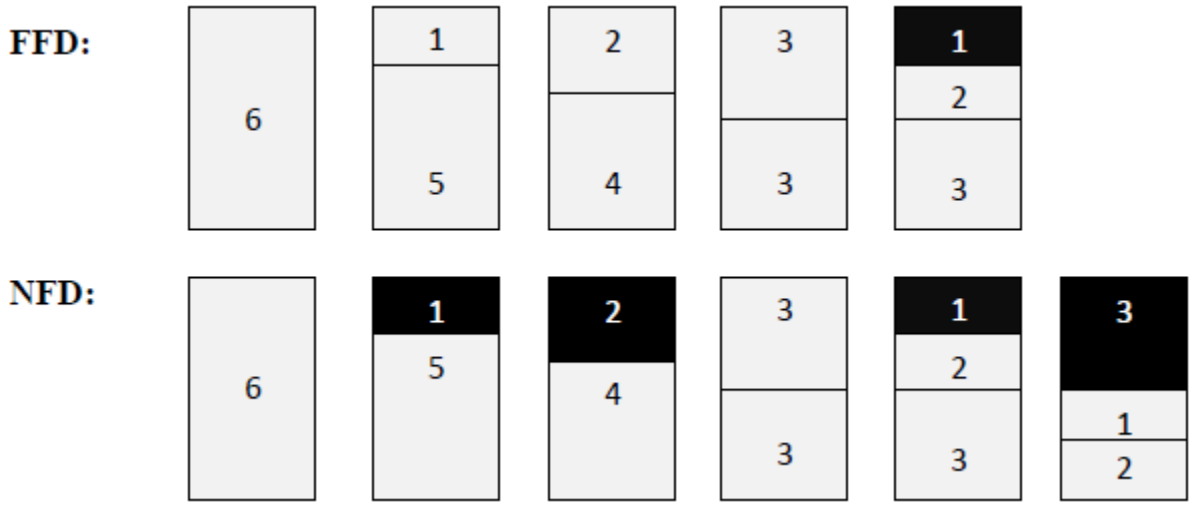
(paketi se smeštaju u otvoreno skladište dok god mogu u njega stati)

Algoritam 2.3: Pseudo kod – NFD

```
Poređaj pakete nerastuće;
Ponavljaj:
        For svaki paket do
            If paket može stati u trenutno otvoreno skladište
                Stavi paket;
            Else
                Otvori novo skladište i smesti paket u njega;
                Zatvori prethodno skladište;
        End for
Sve dok: Nisu svi paketi raspoređeni;
Izlaz:   Paketi raspoređeni u skladišta.
```

PRIMER KORIŠĆENJA GREEDY HEURISTIKA

Neka je dat niz paketa $\mathbb{L} = \{4,1,2,5,3,2,3,6,3\}$ i neka su skladišta kapaciteta 6. Paketi seređaju u nerastućem poretku $\mathbb{L} = \{6,5,4,3,3,3,2,2,1\}$. Nakon toga, različite heuristike smeštaju pakete na različite načine. Od značaja je napomenuti da je teorijski optimum 5.



Metode BFD i WFD su za uzeti primer imale isti rezultat kao FFD.

Dosadašnje metode rešavanje problema BPP -nastavak

- **Heuristike:**

  Greedy (pohlepne) heuristike – FFD, BFD, NFD, WFD, (Dyckhoff , 1990)

  Heuristika redukcije (Reduction Procedure - MTP), Martello and Toth, (1999)

- **Metaheuristike:**

  Hibridni genetski algoritam grupisanja (Hybrid Grouping Genetic Algorithm - HGGA), Falkenauer (1996)

  Optimizacija pomoću mravljih kolonija (Ant Colony Optimization - ACO), Levine i Ducatelle (2004).

- **Predlog nove metode**: Kombinacija metoda ACO i heuristike lokalnog pretraživanja (Local Search -LS), (Rajačić, 2013), po uzoru na Levine i Ducatelle (2004).

Prilagođavanje ACO algoritma problemu BPP

Specifičnost BPP je da u opštem slučaju možemo imati više paketa istih veličina.

Veličina paketa je najvažnije svojstvo, tako da **nije bitan redni broj paketa već njegova veličina.**

Svaki paket možemo identifikovati sa njegovom veličinom: **„paket i“** znači **„paket veličine i“**. Ako veličina paketa nije ceo broj?

Definicija matrice feromona za BPP:

- Element **τ(i,j)** matrice feromona se definiše kao poželjnost da se paket ***i*** i paket ***j*** nađu u istom skladištu.
- Matrica feromona čuva informacije o “dobrim” parovima paketa za smeštanje u isto skladište.
- Dobre veze između paketa se pojačavaju tokom izvršavanja algoritma.

FUNKCIJA PRILAGOĐENOSTI

- **Funkcija prilagođenosti** je indikator kvaliteta rešenja
- U slučaju BPP, vrednost funkcije prilagođenosti za rešenje **s** računa se kao:

$$f(s) = \frac{\sum_{i=1}^{N} (F_i / C)^z}{N}$$

F_i je popunjenost skladišta **i**

C je kapacitet skladišta

N je broj skladišta

z je parametar čija je vrednost određena eksperimentalnim putem (vrednost je postavljena na *z=2)*.

Funkcija prilagođenosti odslikava prosečnu popunjenost skladišta, a ne broj iskorišćenih skladišta

KONSTRUKCIJA REŠENJA

- Svaki mrav počinje sa praznim skladištem i skupom paketa koji treba smestiti.
- Mrav konstruiše rešenje tako što puni skladišta dok svi paketi nisu spakovani.
- U slučaju da nijedan od preostalih paketa ne može stati u trenutno otvoreno skladište, ono se zatvara i otvara se novo.

Koji paket smestiti u otvoreno skladište?

Odluka se zasniva na informaciji iz heuristike, ali i na informaciji iz feromona.

Verovatnoća da će u mrav ***k*** izabrati da paket ***j*** smesti u otvoreno skladište ***b*** nakon konstrukcije parcijalnog rešenja ***s*** je definisana sa:

$$p_k(s,b,j) = \begin{cases} \dfrac{[\tau_b(j)] \times [\mu(j)]^\beta}{\sum_{g \in J_k(s,b)} [\tau_b(g)] \times [\mu(g)]^\beta} & ,ako\ j \in J_k(s,b) \\ 0 & ,inače \end{cases}$$

$J_k(s,b)$ je skup paketa koji mogu stati u trenutno otvoreno skladište **b.** To su paketi koji su ostali nakon konstrukcije parcijalnog rešenja **s,** a koji se mogu smestiti u **b**

$\tau_b(j)$ je vrednost feromona za paket ***j*** koji se nalazi u skladištu ***b***

μ(j) je vrednost heuristike za paket **j**, **β** je parametar koji kontroliše uticaj heuristike

Izgradnja rešenja

Kako izračunati vrednost feromona $\tau_b(j)$?

Vrednost feromona za paket ***j*** koji se nalazi u skladištu ***b*** i predstavlja sumu svih vrednosti matrice feromona za par paketa ***i*** i ***j*** koji se već nalaze skladištu ***b***, podeljena sumom veličina svih paketa koji se trenutno nalaze u skladištu ***b*** .

$$\tau_b(j) = \begin{cases} \frac{\sum_{i \in b} \tau(i,j)}{|b|} & , ako\ je\ b \neq \emptyset \\ 1 & , \quad inače \end{cases}$$

Kako izračunati vrednost heuristike?

Heuristika se vodi idejom da ćemo uvek izabrati veliki paket za smeštanje pre nego mali. Dakle, heuristički odabir paketa je u direktnoj vezi sa njegovom veličinom. Iz tog razloga, definišemo da je **$\mu(j) = j$**

Ažuriranje feromona

- Istovremeno se vrši pojačavanje feromona i evaporacija

- Ideja je da se feromon pojačava svaki put kada se paketi ***i*** i ***j*** nađu zajedno u istom skladištu u najboljem rešenju s_{best}, a da evaporacija bude konstantna
- Koristi se sledeća formula:

$$\tau(i,j) = \rho \times \tau(i,j) + m \times f(s_{best})$$

- ***m =*** broj koji označava koliko puta su se paketi ***i*** i ***j*** našli u istom skladištu u najboljem rešenju
- **ρ =** konstantna stopa evaporacije
- $\mathbf{s}_{best}$ **=** najbolje rešenje u tekućoj iteraciji ili globalno najbolje rešenje

Strategija pojačavanja feromona

Da li pojačavanje feromona dozvoliti samo najboljem mravu?

- Korišćenje samo najboljeg mrava za pojačavanje feromona čini pretragu previše usmerenom. Ako se primenjuje ova strategija, biće podržane samo kombinacije paketa u skladištima koje se često javljaju u dobrim rešenjima.

- Postoji nekoliko načina da se balansira između diversifikacije i usmeravanja pretraživanja. Jedan od načina je odabir između rešenja najboljeg mrava u tekućoj iteraciji s_{Lb} i rešenja globalno najboljeg mrava s_{Gb}.

- Najpogodnije je uvesti parametar γ koji nagoveštava koliko koristimo s_{Lb} dok ne upotrebimo s_{Gb} ponovo.

Lokalno pretraživanje

- U mnogim aplikacijama za rešavanje NP teških problema, ACO je kombinovan sa LS
- Local search vrši pretragu u okolini nekog inicijalnog rešenja, koje dobijamo pomoću ACO
- **Ideja:** Prazni se ***loc*** najmanje punih skladišta, čime je izvestan broj paketa u inicijalnom rešenju oslobođen. Zatim se pokušava sa sledećim zamenama:

  2 smeštena sa **2 slobodna paketa** ili

  2 smeštena sa **1 slobodnim paketom** ili

  1 smešten sa **1 slobodnim paketom.**
- Uslov: skladišta koja nisu oslobođena moraju biti više popunjena nego pre zamene, da bi zamena bila prihvaćena.
- Ostali slobodni paketi koji nisu nigde dodati se ubacuju pomoću neke od postojećih greedy heuristika NFD, FFD, BFD ili WFD.
- Procedura se ponavlja dok se ne popravi funkcija prilagođenosti (ako je moguće).

PRIMER LOKALNOG PRETRAŽIVANJA

Neka je kapacitet skladišta jednak 10 i neka je izabrana heuristika kojom će preostali paketi da se smeštaju FFD. Parametar *loc* u primeru ima vrednost 2.

- *Rešenje pre primene lokalnog pretraživanja*

  | 6 3 | 7 2 | 4 3 | 5 2 | 3 5 | 4 4 |

  U svakom skladištu (između svaka dva znaka |), smešteni su paketi određenih veličina. U prvom skladištu se nalaze paketi veličina 6 i 3. Kapacitet skladišta je 10, broj iskorišćenih skladišta je 6.

  Gubitak u skladištu definišemo kao preostali prostor nakon smeštanja paketa u njega do popunjavanja kapaciteta skladišta. Tako u prvom skladištu imamo gubitak 1, a ukupan gubitak je jednak 12.

- *Otvaramo dva najmanje puna skladišta (to su skladišta 3 i 4) i oslobađamo pakete:*

  Preostala skladišta: **| 6 3 | 7 2 | 3 5 | 4 4 |**

  Slobodni paketi: **5, 4, 3, 2**

Primer realizacije lokalnog pretraživanja (zamena 1 slobodnog sa 1 smeštenim paketom)

- *Vršimo zamenu:*

  Prvo skladište: | **6 3** | ⟶ | **6 4** | Ostalo {5, 3, 3, 2}.

  Drugo skladište: | **7 2** | ⟶ | **7 3** | Ostalo {5, 3, 2, 2}.

  Treće skladište: | **3 5** | ⟶ | **5 5** | Ostalo {3, 3, 2, 2}.

  Četvrto skladište: | **4 4** | ⟶ | **4 4** | Ostalo {3, 3, 2, 2}.

- *Smeštamo ostale pakete pomoću FFD algoritma:*

  Četvrto skladište: | **4 4** | ⟶ | **4 4 2** | Ostalo {3, 3, 2}.

  Novo skladište: {3, 3, 2} ⟶ | **3 3 2** | Ostalo Ø.

- *Konačno rešenje:*

  | **6 4** | **7 3** | **5 5** | **4 4 2** | **3 3 2** |

  Broj iskorišćenih skladišta je 5, a gubitak 2.

- *Ponavljanje procedure:*

  Nema mogućeg napretka; stajemo.

Eksperimentalni rezultati

Testiranja su izvršena na standardnim ORLIB instancama .

80 test instanci, označene sa **un**

n je broj paketa sa veličinama koje su slučajno izabrane u intervalu 20-100

n=120, 250, 500, 1000

Kapacitet skladišta je C=150

Za svako n generisano je M instanci

Eksperimentalni rezultati – podešavanje parametara

Parametar ***br_mrava:***

br_mrava	*u120*	*u250*	*u500*	*u1000*
10	0.057	0.083	0.458	1.285
15	0.068	0.099	0.567	1.659
20	0.078	0.161	1.103	1.457
25	0.083	0.140	0.733	1.862

Tabela 4.1: Vreme izvršavanja programa u zavisnosti od parametra *br_mrava* dato u sekundama.

Parametar ***β:***

β	*u120*	*u250*	*u500*	*u1000*
1	0.077	0.092	0.606	1.516
2	0.065	0.088	1.351	4.329
5	0.083	0.158	12.749	25.574
10	0.104	0.234	38.013	473.703

Tabela 4.2 : Vreme izvršavanja programa u zavisnosti od parametra *β* dato u sekundama.

Parametar γ :

γ	*u120*	*u250*	*u500*	*u1000*
1	0.057	0.083	0.442	1.171
2	0.062	0.094	0.687	1.784
3	0.063	0.099	0.504	1.301
5	0.058	0.099	0.614	1.175

Tabela 4.3 : Vreme izvršavanja programa u zavisnosti od parametra *γ* dato u sekundama.

Eksperimentalni rezultati – podešavanje parametara

Parametar ***ρ:***

ρ	***u120***	***u250***	***u500***	***u1000***
0.10	0.068	0.109	0.474	1.275
0.25	0.063	0.104	0.613	1.124
0.50	0.062	0.110	0.401	1.177
0.75	0.063	0.112	0.453	1.197
0.95	0.062	0.097	0.518	0.986

Tabela 4.4 : Vreme izvršavanja programa u zavisnosti od parametra *ρ* dato u sekundama.

Parametar ***loc :***

loc	***u120***	***u250***	***u500***	***u1000***
2	0.066	0.103	1.050	4.076
3	0.063	0.110	0.586	1.873
4	0.068	0.100	0.482	1.142
5	0.071	0.172	0.626	1.294
10	0.083*	0.174	0.896	1.170

Tabela 4.5 : Vreme izvršavanja programa u zavisnosti o parametra *loc* dato u sekundama.
Vrednosti ostalih parametara su optimalne.
* označava da nije dobijen teorijski optimum

Poređenje heuristika

Iskorišćena skladišta:

	Opt	FFD	NFD	BFD	WFD
u120	48	Opt + 1	Opt + 19	Opt + 1	Opt + 2
u250	99	Opt + 1	Opt + 39	Opt + 1	Opt + 2
u500	198	Opt + 2	Opt + 82	Opt + 2	Opt + 2
u1000	399	Opt + 4	Opt + 164	Opt + 4	Opt + 4

Tabela 4.7 : Broj iskorišćenih skladišta u svakoj heuristici.

Poređenje brzina izvršavanja:

	u120	u250	u500	u1000
FFD	0.531033	0.597954	0.786479	1.098585
NFD	0.608417	0.674830	0.798720	1.142213
BFD	5.284068	5.578800	7.497821	14.670705
WFD	5.163053	5.368559	6.881902	12.545984

Tabela 4.6 : Rezultati upoređivanja vremena izvršavanja heuristika.
Vreme dato u milisekundama.

Efikasnost algoritma u zavisnosti od heuristike

Iskorišćena skladišta:

	Opt	ACO (FFD)	ACO (NFD)	ACO (BFD)	ACO (WFD)
u120	48	Opt	Opt	Opt	Opt
u250	99	Opt + 2	Opt + 2	Opt	Opt
u500	198	Opt	Opt + 3	Opt + 1	Opt + 1
u1000	399	Opt	Opt	Opt + 3	Opt + 2

Tabela 4.9 : Broj skladišta ACO algoritma u zavisnosti od početne informacije.

Poređenje brzina zvršavanja:

	u120	u250	u500	u1000
ACO (FFD)	49.949217	81.303023	648.139368	2843.421072
ACO (NFD)	52.020834	79.765569	491.532009	1089.491660
ACO (BFD)	51.352208	104.329493	814.436775	1201.873814
ACO (WFD)	51.321411	98.105162	855.232495	1147.719506

Tabela 4.8 : Rezultati upoređivanja vremena izvršavanja ACO algoritma. Vreme dato u milisekundama.

Rezultati poređenja brzina izvršavanja : ACO (NFD) < ACO (FFD) < ACO (WFD) < ACO (BFD)

Poređenje sa drugim pristupima

Iskorišćena skladišta:

	u120	*u250*	*u500*	*u1000*
ACO (FFD)	*Opt*	*Opt* + 2	*Opt*	*Opt*
ACO (NFD)	*Opt*	*Opt* + 2	*Opt* + 3	*Opt*
HGGA	*Opt* + 2	*Opt* + 3	*Opt*	*Opt*
MTP	*Opt* + 2	*Opt* + 12	*Opt* + 44	*Opt* + 78

Tabela 4.11 : Broj iskorišćenih skladišta.

Poređenje brzina izvršavanja:

	u120	*u250*	*u500*	*u1000*
ACO (FFD)	0.049949	0.081303	0.648139	2.843421
ACO (NFD)	0.052021	0.079765	0.491532	1.089492
HGGA	3.339587	2.050825	9.977732	52.317715
MTP	1.848113	2.325678	15.090400	69.615887

Tabela 4.10 : Rezultati upoređivanja ACO algoritma sa drugim algoritmima. Vreme dato u sekundama